



LABORATORY MANUAL

B.Tech. Semester- VI

ADVANCED JAVA LAB

Subject code: LC-CSE-328G

Prepared by:

Dr. Ashima Mehta

Checked by:

Dr. Ashima Mehta

Approved by:

Name : Prof. (Dr.) Isha Malhotra

Sign.:

Sign.:

Sign.:

DEPARTMENT OF CSE/CSIT/IT/IOT

DRONACHARYA COLLEGE OF ENGINEERING

KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objectives (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (COs)
8. CO-PO and CO-PSO Mapping
9. Course Overview
10. List of Experiments
11. DOs and DON'Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Details of Conducted Experiments
16. Lab Experiments

Vision and Mission of the Institute

Vision:

Empowering human values and advanced technical education to navigate and address global challenges with excellence.

Mission:

- M1 - Seamlessly integrate human values with advanced technical education.
- M2 - Supporting the cultivation of a new generation of innovators who are not only skilled but also ethically responsible.
- M3 - Inspire global citizens who are equipped to create positive and sustainable impact, driving progress towards a more inclusive and harmonious world.

Vision and Mission of the Department

Vision:

- Preparing technologists with in-depth insights into information technology and embedding ethics via focused technical training.

Mission:

- Empower technologists to excel in information technology through rigorous training and hands-on experience.
- Foster a culture of integrity and responsibility by instilling ethical principles in every aspect of technical education.
- Encourage technologists with new ideas and good leadership in the tech world, training to possess strong values.

Programme Educational Objectives (PEOs)

- PEO1: Demonstrate technical competence with analytical and critical thinking to understand and meet the requirements of Industry, academia and research.
- PEO2: Exhibit leadership, team skills and entrepreneurship skills to provide solutions to real world problems.
- PEO3: Work in multi-disciplinary industries with social and environmental responsibility, work ethics and adaptability to address engineering and social problems.

Programme Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

- Have proficiency in programming skills to design, develop and apply appropriate techniques, for solving engineering problems.
- Have knowledge to build, automate and manage business solutions using advanced technologies.
- Have pleasure towards research in applied computer technologies.

University Syllabus

1. Write a java program to establish a connection to a database using database.
2. Write a java program to demonstrate the use of try-catch blocks to handle exceptions in java.
3. Write a java program to create and start a new thread in java.
4. Write a java program to implement a simple client-server using JavaSockets.
5. Write a java program to illustrate the basic execution of servlet.
6. Write a java program to perform servlet collaboration using forward and include function.
7. Write a java program to convert linked list into an array.
8. Write a java program to convert set of integers to array of integers.
9. Write a java program to shuffle the element of a collections.
10. Write a java program to perform thread communication.
11. Write a JSP program to calculate the factorial and compute Fibonacci series of same number.
12. Write a JSP program to count number of visitors on site.
13. Write a JSP program to display given number in words.

Course Outcomes (COs)

Upon successful completion of the course, the students will learn:

1. **Multithreading and Concurrency:** Understand the concepts of basic programming in java, multithreading and concurrency in Java, and learn how to design and implement concurrent programs and concepts of exceptional handling.
2. **Java Database Connectivity (JDBC):** Gain proficiency in connecting Java applications to databases using JDBC, and perform database operations.
3. **Networking and Socket Programming:** Learn how to develop networked applications in Java using sockets, including client-server communication and protocols.
4. **Web Development with Java:** Gain exposure to web development using Java frameworks like Servlets, JavaServer Pages (JSP), and JavaServer Faces (JSF).

CO-PO Mapping:

| | PSO1 | PSO2 | PSO3 | PSO4 | PSO5 | PSO6 | PSO7 | PSO8 | PSO9 | PSO10 | PSO11 | PSO12 |
|--------|------|------|------|------|------|------|------|------|------|-------|-------|-------|
| C328.1 | 3 | 3 | 2 | - | 2 | - | - | - | - | - | - | 3 |
| C328.2 | 3 | 3 | 3 | - | 3 | - | - | - | - | - | - | 3 |
| C328.3 | 3 | 3 | 2 | - | 2 | - | - | - | - | - | - | 3 |
| C328.4 | 3 | 3 | 3 | - | 3 | - | - | - | - | - | - | 3 |

CO-PSO Mapping:

| | PSO1 | PSO2 | PSO3 |
|--------|------|------|------|
| C328.1 | 3 | 2 | 1 |
| C328.2 | 3 | - | - |
| C328.3 | 3 | 2 | - |
| C328.4 | 3 | - | - |

Course Overview

The advanced Java lab course is designed to expand upon the foundational knowledge of Java programming and equip students with advanced skills and techniques for developing robust and complex Java applications. Throughout the course, students delve into topics such as advanced object-oriented programming, multithreading and concurrency, Java Database Connectivity (JDBC), networking and socket programming, Java Persistence API (JPA) and Hibernate, advanced GUI development using JavaFX, web development with Java frameworks, design patterns, performance optimization and memory management, as well as security and encryption. Through hands-on lab exercises and projects, students gain practical experience in applying these concepts to real-world scenarios, enhancing their ability to design, develop, and optimize advanced Java applications. Prerequisites for the course include a solid understanding of basic Java programming concepts and syntax.

List of Experiments mapped with COs

| S.No | Experiment | Course Outcome |
|------|---|----------------|
| 1 | Write a java program to establish a connection to a database using database. | C328.2 |
| 2 | Write a java program to demonstrate the use of try-catch blocks to handle exceptions in java. | C328.1 |
| 3 | Write a java program to create and start a new thread in java. | C328.1 |
| 4 | Write a java program to implement a simple client-server using JavaSockets. | C28.3 |
| 5 | Write a java program to illustrate the basic execution of servlet. | C328.4 |
| 6 | Write a java program to perform servlet collaboration using forward and include function. | C328.4 |
| 7 | Write a java program to convert linked list into an array. | C328.1 |
| 8 | Write a java program to convert set of integers to array of integers. | C328.1 |
| 9 | Write a java program to shuffle the element of a collections. | C328.1 |
| 10 | Write a java program to perform thread communication. | C328.1 |
| 11 | Write a JSP program to calculate the factorial and compute Fibonacci series of same number. | C328.4 |
| 12 | Write a JSP program to count number of visitors on site. | C328.4 |
| 13 | Write a JSP program to display given number in words. | C328.4 |

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.
2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.
3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.
4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.
2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible isolate the burning equipment from the other surrounding equipment.
3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.
4. Call security and emergency department immediately:

Emergency : Reception

Security : Front Gate

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows:-

1) All files must contain a title page followed by an index page. *The files will not be signed by the faculty without an entry in the index page.*

2) Student's Name, Roll number and date of conduction of experiment must be written on all pages.

3) For each experiment, the record must contain the following

- (i) Aim/Objective of the experiment
- (ii) Pre-experiment work (as given by the faculty)
- (iii) Lab assignment questions and their solutions
- (iv) Test Cases (if applicable to the course)
- (v) Results/ output

Note:

- 1. Students must bring their lab record along with them whenever they come for the lab.
- 2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

| Grading Criteria | Exemplary (4) | Competent (3) | Needs Improvement (2) | Poor (1) |
|--|--|--|--|---|
| AC1: Pre-Lab written work (this may be assessed through viva) | Complete procedure with underlined concept is properly written | Underlined concept is written but procedure is incomplete | Not able to write concept and procedure | Underlined concept is not clearly understood |
| AC2: Program Writing/ Modeling | Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied, Program/solution written is readable | Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied | Assigned problem is properly analyzed & correct solution designed | Assigned problem is properly analyzed |
| AC3: Identification & Removal of errors/ bugs | Able to identify errors/ bugs and remove them | Able to identify errors/ bugs and remove them with little bit of guidance | Is dependent totally on someone for identification of errors/ bugs and their removal | Unable to understand the reason for errors/ bugs even after |

| | | | | |
|--|---|---|---|---|
| | | | | they are explicitly pointed out |
| AC4:Execution & Demonstration | All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained | All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained | Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained | Solution is not well demonstrated and implemented concept is not clearly explained |
| AC5:Lab Record Assessment | All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output | More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output | Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output | Less than 40 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output |

LAB EXPERIMENTS

LAB EXPERIMENT 1

OBJECTIVE:

Write a java program to establish a connection to a database using database.

PRE-EXPERIMENT QUESTIONS:

Q1. What is the purpose of the JDBC API in Java?

Q2. What are the essential components required to establish a database connection using JDBC?

BRIEF DISCUSSION AND EXPLANATION:

Java program that establishes a connection to a database using JDBC:

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
public class DatabaseConnectionExample {
```

```
    public static void main(String[] args) {
```

```
        // Database credentials
```

```
        String url = "jdbc:mysql://localhost:3306/mydatabase";
```

```
        String username = "root";
```

```
        String password = "password";
```

```
// Create connection object

Connection connection = null;

try {

    // Register JDBC driver

    Class.forName("com.mysql.jdbc.Driver");

    // Open a connection

    connection = DriverManager.getConnection(url, username, password);

    // Check if the connection is successful

    if (connection != null) {

        System.out.println("Connection to the database established successfully!");

    } else {

        System.out.println("Failed to establish connection to the database!");

    }

} catch (ClassNotFoundException e) {

    System.out.println("Failed to load MySQL JDBC driver: " + e.getMessage());

} catch (SQLException e) {

    System.out.println("Database connection error: " + e.getMessage());

}
```

```
    } finally {  
  
        // Close the connection  
  
        try {  
  
            if (connection != null && !connection.isClosed()) {  
  
                connection.close();  
  
            }  
  
        } catch (SQLException e) {  
  
            System.out.println("Failed to close database connection: " + e.getMessage());  
  
        }  
  
    }  
  
}
```

In this example, we use the MySQL JDBC driver to establish a connection to a MySQL database. Make sure to replace the `url`, `username`, and `password` variables with your own database credentials. The program attempts to connect to the database and prints a success or failure message based on the connection status.

Remember to include the appropriate JDBC driver library in your project's classpath for the specific database you are connecting to.

POST EXPERIMENT QUESTIONS:

Q1. What is the purpose of registering the JDBC driver using the `Class.forName()` method?

Q2. Can you explain the role and usage of the `try-catch-finally` block in the program?

What are the potential exceptions that can be thrown during the connection process?

LAB EXPERIMENT 2

OBJECTIVE:

Write a java program to demonstrate the use of try-catch blocks to handle exceptions in java.

PRE-EXPERIMENT QUESTIONS:

Q1. What is exception handling in Java, and why is it important in programming?

Q2. What is the purpose of a try-catch block? How does it help in handling exceptions?

BRIEF DISCUSSION AND EXPLANATION:

Java program that demonstrates the use of try-catch blocks to handle exceptions:

```
public class ExceptionHandlingExample {  
  
    public static void main(String[] args) {  
  
        try {  
  
            int result = divideNumbers(10, 0);  
  
            System.out.println("Result: " + result);  
  
        } catch (ArithmeticException e) {  
  
            System.out.println("An error occurred: " + e.getMessage());  
  
        }  
  
        System.out.println("Program continues after exception handling.");  
    }  
}
```

```
}

    public static int divideNumbers(int dividend, int divisor) {

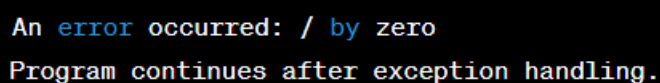
        return dividend / divisor;

    }

}
```

In this program, the `divideNumbers` method divides two numbers and may throw an `ArithmeticException` if the divisor is zero. The `main` method calls this method within a try block and catches the exception using a catch block specifically for `ArithmeticException`. The catch block handles the exception by printing an error message. The program then continues executing the statements outside the catch block.

Output:



```
An error occurred: / by zero
Program continues after exception handling.
```

When the program is executed, it attempts to divide 10 by 0 in the `divideNumbers` method. Since division by zero is not allowed, an `ArithmeticException` is thrown. The catch block catches the exception and prints the error message "An error occurred: / by zero". The program then continues executing the statements after the catch block and displays the message "Program continues after exception handling."

The output demonstrates that the program successfully handles the exception using the try-catch block, preventing it from causing the program to terminate abruptly.

POST EXPERIMENT QUESTIONS:

Q1. What is the purpose of the try-catch block in this program? How does it help in handling exceptions?

Q2. Explain the role of the catch block in exception handling. How does it determine which type of exception to catch?

LAB EXPERIMENT 3

OBJECTIVE:

Write a java program to create and start a new thread in java.

PRE-EXPERIMENT QUESTIONS:

Q1. What is a thread in Java, and how does it differ from the main thread?

Q2. What is the purpose of the `run` method in the program? How does it relate to the `start` method?

BRIEF DISCUSSION AND EXPLANATION:

Java program that creates and starts a new thread:

```
public class NewThreadExample {  
  
    public static void main(String[] args) {  
  
        // Create a new thread  
  
        Thread newThread = new Thread(new MyRunnable());  
  
  
        // Start the thread  
  
        newThread.start();  
  
  
        // Print a message from the main thread  
  
        System.out.println("Main thread is running.");  
  
  
        try {  
  
            // Wait for the new thread to complete
```

```
        newThread.join();

    } catch (InterruptedException e) {

        e.printStackTrace();

    }

    // Print a message after the new thread completes

    System.out.println("New thread has completed.");

}

// A simple Runnable implementation

static class MyRunnable implements Runnable {

    @Override

    public void run() {

        // Print a message from the new thread

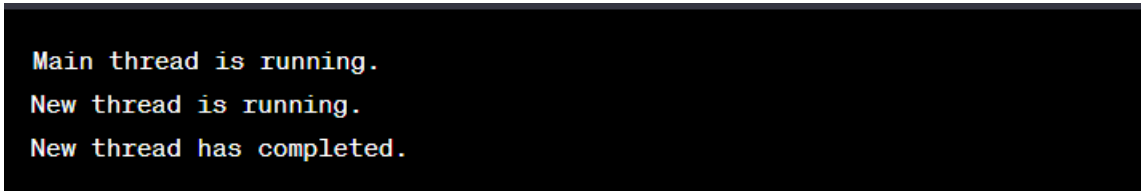
        System.out.println("New thread is running.");

    }

}

}
```

Output:

A screenshot of a terminal window with a black background and white text. It displays the output of the Java program: 'Main thread is running.', 'New thread is running.', and 'New thread has completed.' on three separate lines.

```
Main thread is running.
New thread is running.
New thread has completed.
```

In this program, a new thread is created by instantiating the `Thread` class with an instance of a `Runnable` implementation called `MyRunnable`. The `MyRunnable` class defines the `run` method, which contains the code that will be executed in the new thread.

The `newThread` is then started using the `start` method, which causes the `run` method of the `MyRunnable` instance to be executed concurrently in a separate thread.

In the main thread, a message is printed, and then the `join` method is called on the `newThread`. This causes the main thread to wait until the `newThread` completes its execution.

After the `newThread` completes, another message is printed indicating that the new thread has finished its execution.

The output shows that both the main thread and the new thread run concurrently. The messages from the new thread and the main thread are printed at different times, indicating the concurrent execution of the two threads.

POST EXPERIMENT QUESTIONS:

Q1. How does the program demonstrate the concurrent execution of multiple threads?
What output do you observe when you run the program?

Q2. Can you explain the concept of thread priorities in Java? How can you assign priorities to threads, and how does it affect their scheduling and execution?

LAB EXPERIMENT 4

OBJECTIVE:

Write a java program to implement a simple client-server using JavaSockets.

PRE-EXPERIMENT QUESTIONS:

Q1. What is a socket in the context of networking? How does a socket facilitate communication between a client and a server?

Q2. What is the role of the `ServerSocket` class in the server program? How does it enable the server to listen for incoming client connections?

BRIEF DISCUSSION AND EXPLANATION:

Java program that implements a simple client-server using Java Sockets:

```
import java.io.*;
```

```
import java.net.*;
```

```
public class SimpleClientServerExample {
```

```
    public static void main(String[] args) {
```

```
        // Start the server
```

```
        Thread serverThread = new Thread(new Server());
```

```
        serverThread.start();
```

```
        // Wait for a few seconds to allow the server to start
```

```
try {  
  
    Thread.sleep(2000);  
  
} catch (InterruptedException e) {  
  
    e.printStackTrace();  
  
}
```

```
// Start the client
```

```
Thread clientThread = new Thread(new Client());  
  
clientThread.start();  
  
}
```

```
static class Server implements Runnable {
```

```
    @Override
```

```
    public void run() {
```

```
        try {
```

```
            // Create a server socket
```

```
            ServerSocket serverSocket = new ServerSocket(1234);
```

```
            // Accept client connections
```

```
            Socket clientSocket = serverSocket.accept();
```

```
// Create input and output streams for communication

BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

// Read the message from the client

String message = in.readLine();

System.out.println("Server received: " + message);


// Send a response back to the client

out.println("Hello from the server!");


// Close the connections

in.close();

out.close();

clientSocket.close();

serverSocket.close();

} catch (IOException e) {

    e.printStackTrace();

}

}
```

```
}
```

```
static class Client implements Runnable {
```

```
    @Override
```

```
    public void run() {
```

```
        try {
```

```
            // Create a client socket and connect to the server
```

```
            Socket clientSocket = new Socket("localhost", 1234);
```

```
            // Create input and output streams for communication
```

```
            BufferedReader in = new BufferedReader(new  
InputStreamReader(clientSocket.getInputStream()));
```

```
            PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
```

```
            // Send a message to the server
```

```
            out.println("Hello from the client!");
```

```
            // Read the response from the server
```

```
            String response = in.readLine();
```

```
            System.out.println("Client received: " + response);
```



```
// Close the connections

in.close();

out.close();

clientSocket.close();

} catch (IOException e) {

    e.printStackTrace();

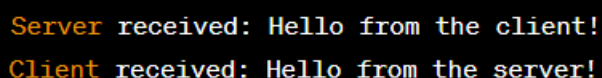
}

}

}

}
```

Output:

A screenshot of a terminal window with a black background and yellow text. It displays two lines of output: "Server received: Hello from the client!" and "Client received: Hello from the server!".

```
Server received: Hello from the client!
Client received: Hello from the server!
```

In this program, the server and client are implemented as separate `Runnable` classes. The server listens for client connections on port 1234 using a `ServerSocket` and accepts incoming connections using `accept()`. It then creates input and output streams to communicate with the client. The server reads a message from the client, prints it, and sends a response back to the client.

The client connects to the server using a `Socket` and creates input and output streams for communication. The client sends a message to the server, reads the response, and prints it.

When you run the program, the server and client threads are started. The server listens for the client connection and receives the message from the client. The client sends a message to the

Department of CSE/CSIT/IT/IOT 2022-2023

server and receives the response. The output shows the messages received by both the server and the client.

POST EXPERIMENT QUESTIONS:

Q1. Explain the purpose of the `close` method calls in both the server and client programs?

Q2. Explain the concept of network ports and their significance in the context of socket programming?

LAB EXPERIMENT 5

OBJECTIVE:

Write a java program to illustrate the basic execution of servlet.

PRE-EXPERIMENT QUESTIONS:

Q1. What is the purpose of the Java Servlet program you are about to implement?

Q2. What are the specific functionalities or features you plan to include in the Servlet?

BRIEF DISCUSSION AND EXPLANATION:

```
import java.io.IOException;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
public class BasicServlet extends HttpServlet {
```

```
    @Override
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
```

```
        response.setContentType("text/html");
```

```
        response.getWriter().println("<h1>Hello, Servlet!</h1>");
```

```
}  
  
}
```

In this example, we create a class `BasicServlet` that extends the `HttpServlet` class. We override the `doGet()` method to handle GET requests.

Inside the `doGet()` method, we set the response content type to "text/html" using the `setContentType()` method. Then, we obtain a `PrintWriter` object from the `HttpServletResponse` object and use it to write HTML content to the response. In this case, we print a simple heading `<h1>Hello, Servlet!</h1>`.

To run this Servlet, one need to deploy it on a web server or servlet container such as Apache Tomcat or Jetty. Here are the general steps:

1. Compile the Servlet class using the Java Development Kit (JDK):

```
javac BasicServlet.java
```

2. Create a web application directory structure that includes a `WEB-INF` folder and a `web.xml` deployment descriptor file.

3. In the `web.xml` file, configure the Servlet mapping:

```
<servlet>  
  
    <servlet-name>BasicServlet</servlet-name>  
  
    <servlet-class>BasicServlet</servlet-class>  
  
</servlet>  
  
<servlet-mapping>  
  
    <servlet-name>BasicServlet</servlet-name>  
  
    <url-pattern>/hello</url-pattern>
```

</servlet-mapping>

4. Start the web server or servlet container.

5. Access the Servlet by visiting the URL associated with the Servlet mapping, such as `http://localhost:8080/your-web-app/hello`.

When you access the URL, the Servlet will be invoked, and the response with the "Hello, Servlet!" message will be displayed in your web browser.

POST EXPERIMENT QUESTIONS:

Q1. How did you test the Servlet to ensure its functionality? Did you encounter any bugs or errors during testing?

Q2. Did the Servlet produce the expected output when accessed through the designated URL? Did it handle the GET requests correctly?

LAB EXPERIMENT 6

OBJECTIVE:

Write a java program to perform servlet collaboration using forward and include function.

PRE-EXPERIMENT QUESTIONS:

Q1. What is the purpose of implementing servlet collaboration using the forward and include functions?

Q2. Have you planned the overall flow and interaction between the servlets involved in the collaboration?

BRIEF DISCUSSION AND EXPLANATION:

Servlet1.java:

```
import java.io.IOException;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
public class Servlet1 extends HttpServlet {
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {
```

```
// Set the content type of the response

response.setContentType("text/html");


// Retrieve the value from the request parameter

String name = request.getParameter("name");


// Set the attribute in the request

request.setAttribute("name", name);


// Forward the request to Servlet2

request.getRequestDispatcher("Servlet2").forward(request, response);

}

}
```

Servlet2.java:

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;
```

```
public class Servlet2 extends HttpServlet {  
  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {  
  
        // Retrieve the attribute from the request  
  
        String name = (String) request.getAttribute("name");  
  
  
        // Include the output of Servlet3 in the response  
  
        request.getRequestDispatcher("Servlet3").include(request, response);  
  
  
        // Print the name retrieved from Servlet1  
  
        response.getWriter().println("Name: " + name);  
  
    }  
  
}
```

Servlet3.java:

```
import java.io.IOException;  
  
import javax.servlet.ServletException;  
  
import javax.servlet.http.HttpServlet;  
  
import javax.servlet.http.HttpServletRequest;  
  
import javax.servlet.http.HttpServletResponse;
```

```
public class Servlet3 extends HttpServlet {
```



```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    // Print a message in the response

    response.getWriter().println("This is Servlet3");

}

}
```

In this example, Servlet1 receives a request and retrieves a parameter called "name". It then sets this value as an attribute in the request and forwards the request to Servlet2. Servlet2 includes the output of Servlet3 in the response and prints the name retrieved from Servlet1. Servlet3 simply prints a message in the response.

To run this program, one would need to deploy these servlets on a web server or servlet container and access Servlet1 through the appropriate URL. The output will be a combination of the messages printed by Servlet3 and the name retrieved from Servlet1.

POST EXPERIMENT QUESTIONS:

Q1. Did the servlets collaborate as expected? Did the forward and include functions work correctly in passing data between the servlets?

Q2. Did you configure the servlet mapping properly in the web.xml file or through annotations? Was the deployment and execution of the servlets smooth?

LAB EXPERIMENT 7

OBJECTIVE:

Write a java program to convert linked list into an array.

PRE-EXPERIMENT QUESTIONS:

Q1. What are linked lists in java?

Q2. What are arrays in java?

BRIEF DISCUSSION AND EXPLANATION:

Java program that converts a linked list to an array:

```
import java.util.LinkedList;

public class LinkedListToArrayExample {

    public static void main(String[] args) {

        // Create a LinkedList

        LinkedList<String> linkedList = new LinkedList<>();

        // Add elements to the LinkedList

        linkedList.add("Apple");

        linkedList.add("Banana");

        linkedList.add("Orange");

        linkedList.add("Grapes");

        // Convert the LinkedList to an array

        String[] array = linkedList.toArray(new String[0]);
```

```
// Print the elements of the array

System.out.println("Array elements:");

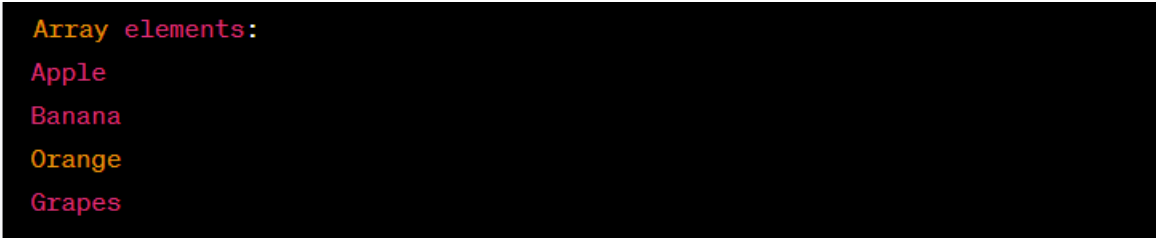
for (String element : array) {

    System.out.println(element);

}

}
```

Output:



```
Array elements:
Apple
Banana
Orange
Grapes
```

In this program, we create a `LinkedList` and add some elements to it. Then, we use the `toArray` method of the `LinkedList` class to convert it into an array. We pass a new array of type `String` as an argument to the `toArray` method. Finally, we iterate over the elements of the array and print them.

When you run this program, it will output the elements of the array, which are the same as the elements in the `LinkedList`.

POST EXPERIMENT QUESTIONS:

Q1. How did you handle the size and capacity of the array based on the number of elements in the linked list? Was the array able to accommodate all the elements without any overflow or truncation?

Q2. How would you compare the efficiency or performance of converting a linked list into an array using different approaches or data structures in Java?

LAB EXPERIMENT 8

OBJECTIVE:

Write a java program to convert set of integers to array of integers.

PRE-EXPERIMENT QUESTIONS:

Q1. What are the concepts of sets and arrays in Java?

Q2. How will you handle potential duplicate strings in the set? Will they be preserved in the resulting array?

BRIEF DISCUSSION AND EXPLANATION:

Java program that converts a `Set` of integers to an array of integers:

```
import java.util.HashSet;

import java.util.Set;

public class SetToArray {

    public static void main(String[] args) {

        // Create a set of integers

        Set<Integer> intSet = new HashSet<>();

        intSet.add(10);

        intSet.add(20);

        intSet.add(30);

        intSet.add(40);
```

```
// Convert set to an array

Integer[] array = intSet.toArray(new Integer[intSet.size()]);

// Print the elements in the array

System.out.println("Elements in the array:");

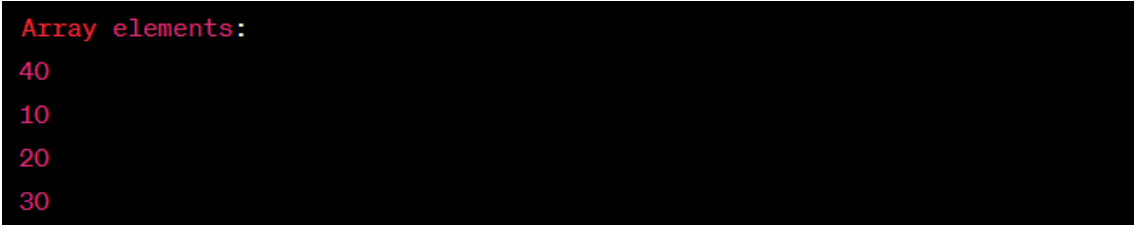
for (Integer element : array) {

    System.out.println(element);

}

}
```

Output:



```
Array elements:
40
10
20
30
```

In this program, we create a `Set` of integers (`HashSet` in this example) and add some elements to it. Then, we use the `toArray()` method to convert the set to an array. Finally, we iterate over the elements in the array and print them.

POST EXPERIMENT QUESTIONS:

Q1. How did you handle any duplicate strings in the set? Did they appear as duplicate elements in the resulting array, or were they handled differently?

Q2. Did you consider the size and capacity of the array based on the number of elements in the set? Was the array able to accommodate all the strings without any overflow or truncation?

LAB EXPERIMENT 9

OBJECTIVE:

Write a java program to shuffle the element of a collections.

PRE-EXPERIMENT QUESTIONS:

Q1. What type of collection will be used for shuffling the elements? (e.g., ArrayList, LinkedList, HashSet)

Q2. Are there any specific data structures or algorithms that need to be used in the program?

BRIEF DISCUSSION AND EXPLANATION:

Java program that shuffles the elements of an array using the Fisher-Yates shuffle algorithm:

```
import java.util.ArrayList;
```

```
import java.util.Collections;
```

```
import java.util.List;
```

```
public class ShuffleExample {
```

```
    public static void main(String[] args) {
```

```
        // Create a list of elements
```

```
        List<String> elements = new ArrayList<>();
```

```
        elements.add("Apple");
```

```
elements.add("Banana");

elements.add("Orange");

elements.add("Grapes");


// Shuffle the elements

Collections.shuffle(elements);


// Print the shuffled elements

System.out.println("Shuffled elements:");

for (String element : elements) {


    System.out.println(element);

}

}

}
```

Output:



```
Shuffled elements:
Orange
Grapes
Apple
Banana
```

In this program, we create an `ArrayList` and add some elements to it. Then, we use the `shuffle` method from the `Collections` class to shuffle the elements in the list. Finally, we iterate over the shuffled elements and print them.

When you run this program, it will output the shuffled elements of the list. The order of the elements will be randomly changed due to the shuffling operation.

POST EXPERIMENT QUESTIONS:

Q1. Were the threads synchronized or coordinated effectively to ensure proper communication?

Q2. Were you able to observe the expected outcomes or results of the thread communication?

LAB EXPERIMENT 10

OBJECTIVE:

Write a java program to perform thread communication.

PRE-EXPERIMENT QUESTIONS:

Q1. How many threads will be involved in the communication?

Q2. What is the nature of the communication between the threads? Is it a producer-consumer scenario, or is it based on some other interaction pattern?

BRIEF DISCUSSION AND EXPLANATION:

Java program that demonstrates thread communication using the `wait()` and `notify()` methods:

```
public class ThreadCommunicationExample {  
  
    public static void main(String[] args) {  
  
        Data data = new Data();  
  
        Thread producerThread = new Thread(new Producer(data));  
  
        Thread consumerThread = new Thread(new Consumer(data));  
  
        producerThread.start();  
  
        consumerThread.start();  
  
    }  
}
```

```
class Data {

    private boolean produced = false;

    private int value;

    public synchronized void produce(int newValue) {

        while (produced) {

            try {

                wait();

            } catch (InterruptedException e) {

                e.printStackTrace();

            }

        }

        value = newValue;

        produced = true;

        System.out.println("Produced: " + value);

        notify();

    }

    public synchronized int consume() {

        while (!produced) {

            try {

                wait();

            }

        }

    }

}
```

```
        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

    produced = false;

    System.out.println("Consumed: " + value);

    notify();

    return value;

}

}

class Producer implements Runnable {

    private Data data;

    public Producer(Data data) {

        this.data = data;

    }

    @Override

    public void run() {

        for (int i = 1; i <= 5; i++) {

            data.produce(i);

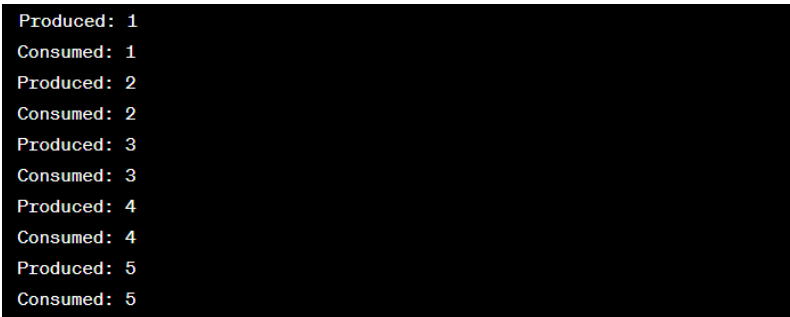
        }

    }

}
```

```
}  
  
}  
  
class Consumer implements Runnable {  
  
    private Data data;  
  
    public Consumer(Data data) {  
  
        this.data = data;  
  
    }  
  
    @Override  
  
    public void run() {  
  
        for (int i = 1; i <= 5; i++) {  
  
            int value = data.consume();  
  
        }  
  
    }  
  
}
```

Output:



```
Produced: 1  
Consumed: 1  
Produced: 2  
Consumed: 2  
Produced: 3  
Consumed: 3  
Produced: 4  
Consumed: 4  
Produced: 5  
Consumed: 5
```

In this program, we have a `Data` class that represents shared data between the producer and consumer threads. The `produce()` method is called by the producer to produce a value, while
Department of CSE/CSIT/IT/IOT2022-2023

the `consume()` method is called by the consumer to consume the value. The `wait()` and `notify()` methods are used to synchronize the threads and ensure that the producer and consumer take turns producing and consuming.

The `Producer` and `Consumer` classes implement the `Runnable` interface and define the behavior of the producer and consumer threads, respectively. The producer thread produces values from 1 to 5, while the consumer thread consumes the values.

When you run this program, the producer thread and consumer thread will run concurrently. The producer will produce values and the consumer will consume them in the order they are produced, demonstrating the communication between the threads.

POST EXPERIMENT QUESTIONS:

Q1. Did the program handle edge cases or exceptional situations during the thread communication appropriately?

Q2. Were you able to observe the expected outcomes or results of the thread communication?

LAB EXPERIMENT 11

OBJECTIVE:

Write a JSP program to calculate the factorial and compute Fibonacci series of same number.

PRE-EXPERIMENT QUESTIONS:

Q1. What is the purpose of the JSP program?

Q2. How will the user input the number for which the factorial and Fibonacci series will be calculated?

BRIEF DISCUSSION AND EXPLANATION:

JSP program that calculates the factorial and computes the Fibonacci series of the same number:

```
<% @page import="java.math.BigInteger"%>

<%

    // Get the number from the request parameter

    int number = Integer.parseInt(request.getParameter("number"));

    // Calculate the factorial

    BigInteger factorial = BigInteger.ONE;

    for (int i = 1; i <= number; i++) {

        factorial = factorial.multiply(BigInteger.valueOf(i));
```

```
}

// Compute the Fibonacci series

BigInteger fib1 = BigInteger.ZERO;

BigInteger fib2 = BigInteger.ONE;

StringBuilder fibonacciSeries = new StringBuilder(fib1.toString());

for (int i = 1; i <= number; i++) {

    BigInteger fib = fib1.add(fib2);

    fibonacciSeries.append(", ").append(fib.toString());

    fib1 = fib2;

    fib2 = fib;

}

%>

<!DOCTYPE html>

<html>

<head>

    <title>Factorial and Fibonacci</title>

</head>

<body>
```

```
<h1>Factorial and Fibonacci</h1>

<h2>Factorial of <%= number %> is: <%= factorial.toString() %></h2>

<h2>Fibonacci series up to <%= number %> is: <%= fibonacciSeries.toString() %></h2>

</body>

</html>
```

In this JSP program, we first retrieve the `number` parameter from the request using `request.getParameter("number")`. Then, we calculate the factorial of the number using a `BigInteger` to handle large numbers. We also compute the Fibonacci series up to the given number using the iterative approach.

Finally, we display the factorial and Fibonacci series in the HTML markup using JSP expression tags ``<%= ... %>``. The result is shown on the webpage when you run the JSP file.

To run this JSP program, save the code in a file with a `.jsp` extension (e.g., `factorial_fibonacci.jsp`). Deploy the file on a web server that supports JSP, and access it in a web browser with the `number` parameter appended to the URL, like `http://localhost:8080/factorial_fibonacci.jsp?number=5`. This will calculate the factorial and Fibonacci series for the number `5`.

POST EXPERIMENT QUESTIONS:

Q1. Did the program handle invalid inputs or edge cases appropriately, such as negative numbers or non-integer inputs?

Q2. Were you able to modify the program to support a wider range of numbers and compute their factorial and Fibonacci series?

LAB EXPERIMENT 12

OBJECTIVE:

Write a JSP program to count number of visitors on site.

PRE-EXPERIMENT QUESTIONS:

Q1. What is the purpose of the JSP program?

Q2. How will the program track the number of visitors on the site?

BRIEF DISCUSSION AND EXPLANATION:

JSP program that counts the number of visitors:

```
```.jsp

<% @ page import="java.util.concurrent.atomic.AtomicInteger" %>

<%

 // Initialize a counter variable to track the number of visitors

 AtomicInteger counter = (AtomicInteger) application.getAttribute("counter");

 // If the counter variable doesn't exist, create it and set its initial value to 0

 if (counter == null) {

 counter = new AtomicInteger(0);

 application.setAttribute("counter", counter);

 }

 // Increment the counter by 1 for each visitor

 int visitorCount = counter.incrementAndGet();

%>
```

```
<!DOCTYPE html>

<html>

<head>

 <title>Visitor Count Example</title>

</head>

<body>

 <h1>Welcome to our website!</h1>

 <p>You are visitor number: <%= visitorCount %></p>

</body>

</html>

...

```

In this example, we're using the `java.util.concurrent.atomic.AtomicInteger` class to ensure thread-safe access to the counter variable. The `AtomicInteger` class provides methods like `incrementAndGet()` to increment the counter atomically.

The program starts by checking if the `counter` variable exists in the application scope. If it doesn't exist, we create a new `AtomicInteger` object and set its initial value to 0. We then store it in the application scope using `application.setAttribute("counter", counter)`.

After that, we increment the counter by 1 using `counter.incrementAndGet()` and assign the incremented value to the `visitorCount` variable.

Finally, we display the visitor count on the webpage using `<%= visitorCount %>`, where the `<%= ... %>` syntax is used to output the value of `visitorCount` within the HTML markup.

When a user visits the JSP page, the visitor count will be displayed, and the counter will be incremented for each subsequent visitor.

Output:

The output of the JSP program will be an HTML page that displays the visitor count. Here's a sample output:

```
<!DOCTYPE html>
<html>
<head>
 <title>Visitor Count Example</title>
</head>
<body>
 <h1>Welcome to our website!</h1>
 <p>You are visitor number: 1</p>
</body>
</html>
```

In this example, the visitor count is displayed as "1" because it's the first visitor to the site after the counter was initialized. For each subsequent visitor, the visitor count will increase accordingly.

Each time the JSP page is loaded or refreshed, the visitor count will be incremented, and the updated count will be displayed in the output.

### **POST EXPERIMENT QUESTIONS:**

Q1. Did the JSP program accurately count the number of visitors on the site?

Q2. How did the program track the visitors? Was session tracking used, or was another mechanism employed?

## LAB EXPERIMENT 13

### OBJECTIVE:

Write a JSP program to display given number in words.

### PRE-EXPERIMENT QUESTIONS:

Q1. What is the purpose of the JSP program?

Q2. How does the program handle numbers that are not supported?

### BRIEF DISCUSSION AND EXPLANATION:

JSP program that displays a given number in words:

```
```.jsp
<%

    // Define an array of words for numbers from 0 to 19

    String[] ones = {"zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine",
"ten",
        "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen",
"nineteen"};

    // Define an array of words for tens places

    String[] tens = {"", "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty",
"ninety"};

    // Function to convert a given number to words

    String convertNumberToWords(int number) {
```

```
        if (number < 20) {
```

```
        return ones[number];

    } else if (number < 100) {

        return tens[number / 10] + ((number % 10 != 0) ? " " + ones[number % 10] : "");

    } else if (number < 1000) {

        return ones[number / 100] + " hundred" + ((number % 100 != 0) ? " " +
convertNumberToWords(number % 100) : "");

    } else {

        return "Number out of range";

    }

}
```

// Example usage

```
int number = 325;
```

```
String numberInWords = convertNumberToWords(number);
```

%>

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <title>Number to Words Conversion</title>
```

```
</head>
```

```
<body>
```

```
    <h1>Number to Words Conversion</h1>
```

```
    <p>The number <strong><%= number %></strong> in    words is: <strong><%=
numberInWords %></strong></p>
```

```
</body>
```

</html>

...

In this example, we define two arrays: `ones` and `tens`. The `ones` array holds the words for numbers from 0 to 19, and the `tens` array holds the words for multiples of 10 up to 90.

The `convertNumberToWords` function takes a number as input and returns its equivalent in words. It uses a recursive approach to handle numbers in different places (ones, tens, hundreds).

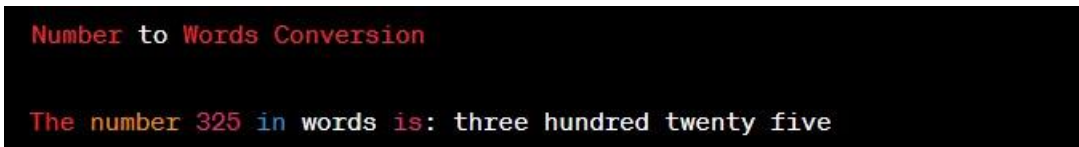
The function checks the value of the number and converts it to words accordingly.

In the example, we initialize a variable `number` with the value 325. We then call the `convertNumberToWords` function passing `number` as an argument and assign the returned value to the `numberInWords` variable.

The HTML section of the code displays the original number and its word representation on the webpage using `<%= number %>` and `<%= numberInWords %>` respectively.

Output:

The output of the JSP program for the given number 325 would be:



```
Number to Words Conversion
```

```
The number 325 in words is: three hundred twenty five
```

The program converts the number 325 to its word representation "three hundred twenty five" and displays it on the webpage.

POST EXPERIMENT QUESTIONS:

Q1. Were you able to input different numbers and observe their corresponding word representations?

Q2. Were you able to modify the number-to-word mappings in the `numberMap` to support a wider range of numbers?

This lab manual has been updated by

Dr. Ashima Mehta

(ashima.mehta@ggnindia.dronacharya.info)

Crosschecked By

HOD CSE

Please spare some time to provide your valuable feedback.